Using Buildbot

Adding New Tests

TODO: detailed description. Short version: add new tests that run from "make check".

Adding New Build Types

Sometimes, just adding tests isn't enough, and you need to add completely new types of builds using different commands or different configuration options. Doing so is straightforward and not much more complicated than typing out the shell commands you want to run. Let's start with a simple example:

```
moab_serial = factory.BuildFactory()
moab_serial.addSteps([
        Clean(),
        SVN(svnurl=moaburl),
        Autoreconf(),
        Configure(),
        Compile(),
        Test(command=["make", "check"]),
        ])

c['builders'].append({
        'name': "moab-serial",
        'slavename': "gnep",
        'builddir': "moab-serial",
        'factory': moab_serial,
        'category': "moab",
        })
```

The first thing we do is create a `BuildFactory`, which will hold the steps we want to run in our build. We then add the list of steps we'd like to run. Each step is a Python object that we construct in-place. Buildbot ships with a bunch of commands (e.g. `SVN`, `Compile`, `Test`), and our Buildbot adds a few more (`Clean`, `Autoreconf`, `Configure`, `Distcheck`, `Install`, `Upload`). Some of these steps have special options you can pass to the constructor to change their behavior.

You might notice that we specify a command for the `Test` step; this changes the shell command that gets run. You may also notice that we split the command into two separate strings. This allows the Buildbot to bypass the shell and execute the command directly.

Once we've added our build steps, we can then create the builder itself. To do so, we simply add some configuration options to a list of builders, specifying a name, slave(s) to run the build on, the directory to build in, the factory we just created, and optionally a category. We use the category to separate out the builds so that MOAB emails go to the MOAB list, CGM emails go to the CGM list, etc.

Not let's look at a more advanced example:

```
pytaps = factory.BuildFactory()
pytaps.addSteps([
        SVN(svnurl=pytapsurl),
        Compile(command=[
                "python", "setup.py",
                    WithProperties("--iMesh-path=%(packages)s/moab-shared"),
                    WithProperties("--iGeom-path=%(packages)s/cgm-shared"),
                    WithProperties("--iRel-path=%(packages)s/lasso-shared"),
                    "install", WithProperties("--home=%(packages)s/pytaps")
                    ]),
        Test(command=["python", "setup.py", "test"],
             env={"LD_LIBRARY_PATH": WithProperties(
                    "%(packages)s/moab-shared/lib:" +
                    "%(packages)s/cgm-shared/lib:" +
                    "%(packages)s/lasso-shared/lib:" +
                    cubit_base+"/bin"
                    )}
             )
        ])
```

Here, our commands are considerably more complicated, using a new construct called `WithProperties`. Each build or build slave can have a number of special properties that you can use in your build steps. Here, we use a property called "packages" which refers to the root directory of our installed libraries. This allows us to run the command on any machine without special-casing the paths for each slave.